REMARKS

The present application was filed on February 8, 2002 with claims 1-12. Claims 1-12 are currently pending in the application. Claims 1, 11 and 12 are the independent claims.

In the Office Action, claims 1-12 are rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6,848,097 to Alverson et al. (hereinafter "Alverson") in view of U.S. Patent No. 6,615,368 to Dunlap (hereinafter "Dunlap").

Applicants note that the Examiner states that the §101 rejection of claims 1-10 made in the prior Office Action "is withdrawn in view of applicant's amendment" (Current Office Action, p. 2). This is not accurate. No amendments were made to claims 1-10 as a result of the prior Office Action, nor at any other time. However, Applicants did provide remarks as to why claims 1-10 should be allowed over the prior §101 rejection.

With respect to the current §103(a) rejection, Applicants respectfully traverse. Applicants request reconsideration of the claims in view of the following remarks.

For a valid §103(a) rejection, the combined references must teach or suggest all the claim limitations. Manual of Patent Examining Procedure (MPEP), Eighth Edition, August 2001, §2143. Applicants respectfully submit that this requirement has not been met with respect to claim 1.

Claim 1 sets forth:

> A method of implementing a software breakpoint in a multiprocessor system having a plurality of processors each coupled to a main memory, each of the processors having an instruction cache associated therewith, the method comprising the steps of:
>
> retrieving an instruction, for which the breakpoint is to be inserted, from a corresponding instruction address in the main memory;
>
> inserting a breakpoint code at the instruction address in main memory; and
>
> after the breakpoint code is executed by a given one of the processors, storing the retrieved instruction in the corresponding instruction cache for that processor and setting a use-once indicator associated with the instruction as stored in the corresponding instruction cache for that processor, wherein the use-once indicator, when set for the instruction as stored in the instruction cache, is operative via cache control logic to clear a validity indicator associated with the instruction after a single fetch of the instruction from the instruction cache, such that subsequent attempts by the given processor to access the instruction as stored in the instruction cache will cause the processor to retrieve the breakpoint code at the instruction address in main memory.

In an illustrative embodiment in accordance with claim 1, the invention is implemented using a debugger which communicates with a multiprocessor system via an interface. The debugger saves a retrieved instruction that has been replaced with a breakpoint code, and when the breakpoint code is executed by a given one of the processors, a breakpoint is taken and the debugger takes control of the given processor. Before the debugger resumes execution, it stores the retrieved instruction in the instruction cache for the given processor, and sets a use-once indicator associated with the instruction as stored in the corresponding instruction cache for that processor. The use-once indicator, when set for the instruction as stored in the instruction cache, is operative via cache control logic to clear a validity indicator associated with the instruction after a single fetch of the instruction from the instruction cache. As a result, subsequent attempts by the given processor to access the instruction as stored in the instruction cache will cause the processor to retrieve the breakpoint code at the instruction address in main memory.

Advantageously, the software breakpoint techniques of the present invention ensure that a specified debug opcode or other breakpoint code can remain present in the main memory of the multiprocessor system at all times, such that all of the processors of the system will reliably fetch and execute that breakpoint code even if one or more of these processors are resuming execution from the breakpoint.

In formulating the §103(a) rejection of claim 1, the Examiner argues that some elements contained in the portion of the claim beginning with the words "after the breakpoint code is executed" are described by Alverson at col. 15, lines 45-54 (Office Action, p. 3). Applicants disagree. The above-cited portion of Alverson states:

> The routine continues at step 510 to determine whether the request is a request from the debugger to create a breakpoint at a specified instruction within the executable code of the target. If so, the routine continues to step 515 to allocate memory for the instruction group to be generated, and then invokes the Generate OOL Instruction Emulation Group subroutine 515 for the instruction to be replaced. The routine continues at step 520 where the generated instruction group is installed in the allocated memory, and information about the created group is stored in an accessible location.

One skilled in the art will recognize that this portion of Alverson describes the replacement of a specified instruction within the executable code with a breakpoint instruction. Obviously, this type of replacement step must occur before the breakpoint can be executed. In claim 1, on the other hand, the steps following the words "after the breakpoint is executed" explicitly occur after a breakpoint instruction is encountered within the executable code. Consequently, Alverson and claim 1 describe entirely different steps. Alverson describes the configuration of breakpoint instructions, while the last clause of claim 1 describes steps that are performed after breakpoints are actually executed. Therefore Alverson fails to teach or suggest these elements of claim 1. Moreover, Dunlap does not correct this fundamental deficiency in Alverson. Accordingly, not all elements of claim 1 are taught or suggested by the proposed reference combination.

In addition, the Examiner further argues that Dunlap's change of flow (COF) flag describes the use-once indicator and the validity indicator in claim 1 (Office Action, p. 4, citing Dunlap at col. 6 lines 42-44 and 46-48). Applicants again respectfully disagree. Dunlap at col. 6, lines 42-48 states:

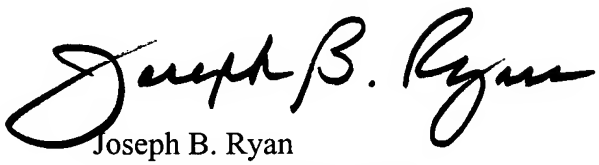
> COF address register 330 detects the active COF flag and loads the next instruction address (in the new branch) for subsequent serial output to external devices (process step 375).
>
> In addition, the next instruction address is stored in IP 340 and is sent to instruction decode/dispatch logic 200, which uses it to fetch the next instruction from the new branch.

Applicants suggest that this portion of Dunlap, nor any other portion for that matter, fails at least to describe the portion of claim 1 wherein the use-once indicator "is operative via cache control logic to clear a validity indicator associated with the instruction after a single fetch of the instruction from the instruction cache." Accordingly, this element of claim 1 is also not taught or suggested by the Alverson-Dunlap reference combination.

Dependent claims 2-10 are believed to be allowable for at least the reasons stated above with respect to claim 1. Independent claims 11 and 12 are system and computer program product claims, respectively, corresponding to method claim 1. Therefore, they are also believed to be allowable for reasons similar to those stated above.

In view of the above, Applicants believe that claims 1-12 are in condition for allowance and respectfully request withdrawal of the §103(a) rejection.

Respectfully submitted,



Date: September 29, 2005

Joseph B. Ryan
Attorney for Applicant(s)
Reg. No. 37,922
Ryan, Mason & Lewis, LLP
90 Forest Avenue
Locust Valley, NY 11560
(516) 759-7517